Spécifier et vérifier des systèmes temps réel avec Fiacre/TINA

Bernard Berthomieu Didier le Botlan
Silvano Dal Zilio François Vernadat

Vertics / LAAS / CNRS / INSA

Journée AFSEC Concurrence
ENS-Cachan, 19 Novembre 2013

Processus de vérification

3 étapes :

Modélisation

$\Rightarrow$ modèle formel M de l'application
$\Rightarrow$ propriétés attendues P, dans la logique L

Abstraction

de M
préservant les formules de L
$\Rightarrow$ graphe d'états abstraits fini A

Vérification

des formules P sur A (Model-Checking)
$\Rightarrow$ VRAI ou un contre-exemple

Avec TINA

Modèles formels

Réseaux de Petri Temporels + Priorités, Données, Chronomètres

Descriptions de haut niveau en FIACRE (compilées)

Abstractions

Abstractions “ordre partiel” pour Réseaux de Petri

Graphes de classes pour Réseaux temporels

Vérification

State/Event LTL (natif)

Mu-Calcul (natif)

exportation abstraction vers outils CADP, MEC

Réseaux de Petri – Abstractions

Abstractions

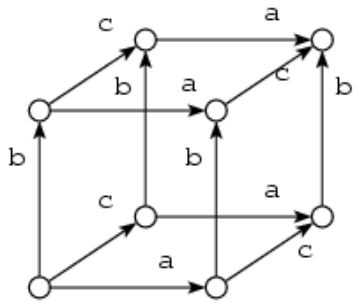
- Graphes de couverture (bornes sup, détectent places non bornées)
- Graphes des marquages (espace d'états exact)
- Réductions ordre partiel
  - Ensembles persistants (blocages)
  - Pas couvrants (blocages+)
  - Pas persistants (blocages+)
- Analyse structurelle
  - Surapproximations par ensembles semi-linéaires

Aussi ...

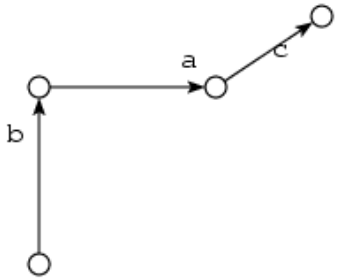
- Méthodes symboliques (SDD)

Réductions ordre partiel

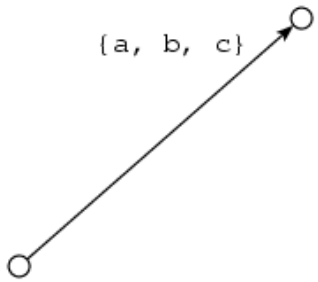
Explorations implicites



Exhaustif :
2^n états, $n \times 2^{n-1}$ transitions

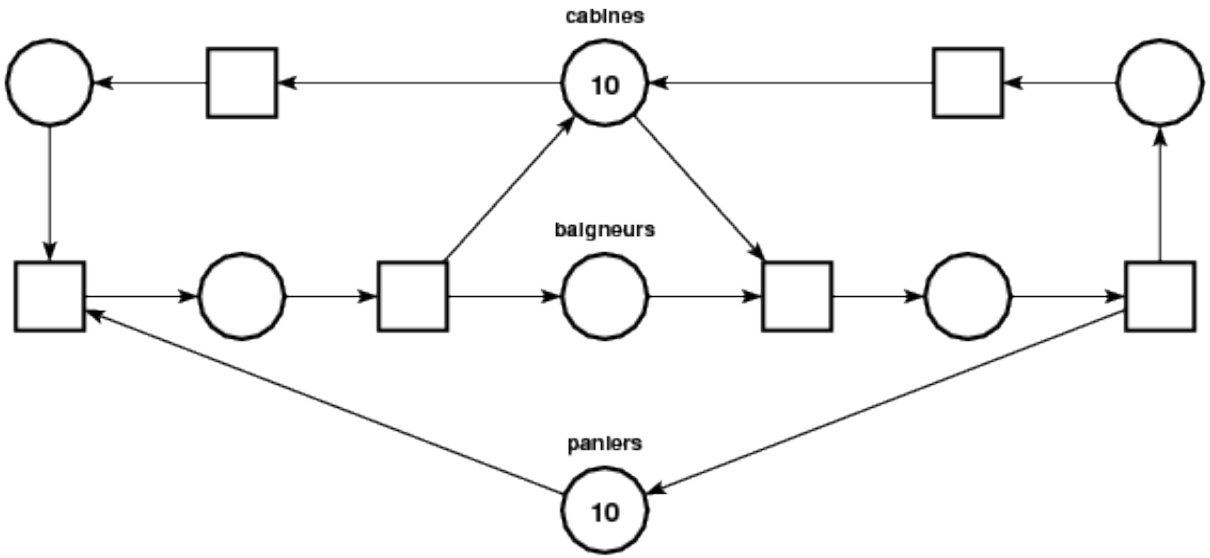


Ensembles persistants :
$(n+1)$ états, n transitions



Graphes de pas couvrants :
2 états, 1 transitions

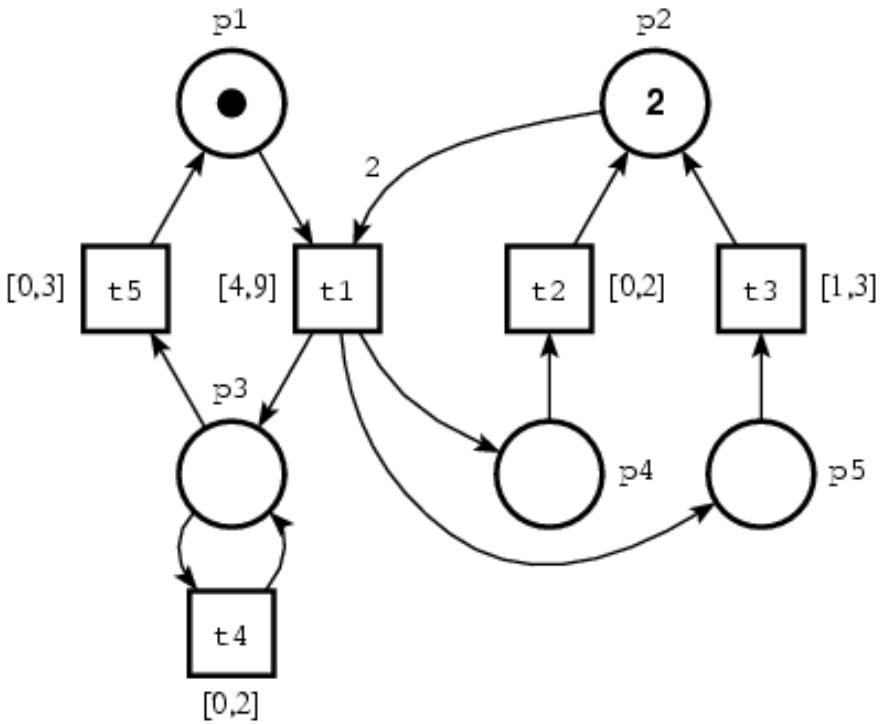
Piscine



		Exact tina -R	*Pas couvrants* tina -V	*Ens persistants* tina -P	*Pas persistants* tina -Q
$K = 10$	*Etats*	7006	367	97	87
	s	0	0	0	0
$K = 100$	*Etats*	$\sim 280M$	39517	997	897
	s	8800	0.3	0	0
$K = 1000$	*Etats*	?	$\sim 4M$	9997	8997
	s	?	30	0	0
$K = 10000$	*Etats*	?	$\sim 400M$	99997	89997
	s	?	4500	0.5	0.5

Réseaux Temporels (Merlin 74)

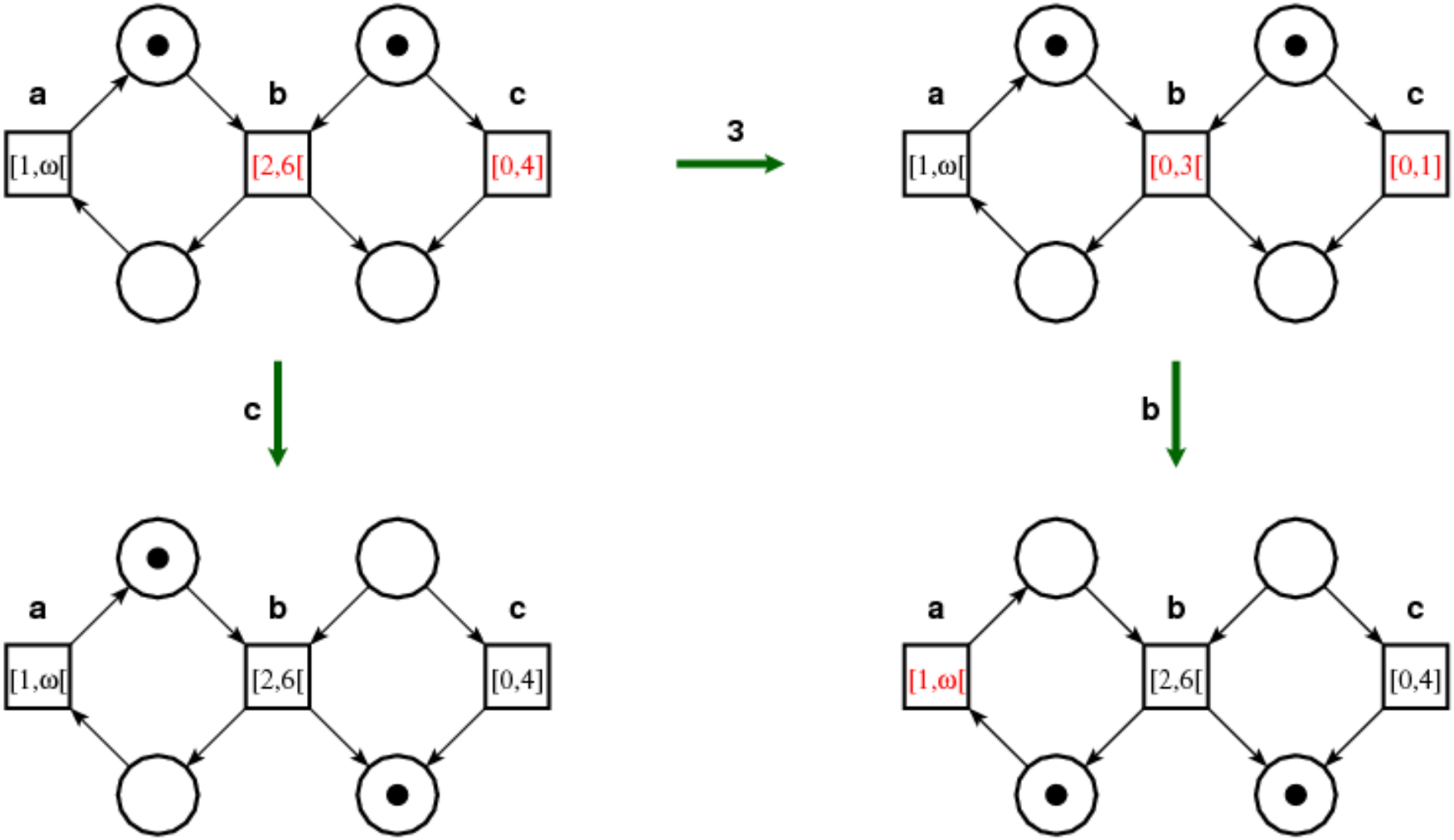
RdP + Intervalles temporels



Espaces d'états infinis (temps dense)

Caractère borné indécidable (mais conditions suffisantes)

États et transitions d'états



TPNs – Abstractions

Abstractions : Graphes de classes d'états

Classe d'état = ensemble d'état

classe = marquage (état discret) + polyèdre (information temporelle)

Differentes constructions, préservant :

Marquages + traces (LTL) (SCG [BM83] [BD91])

Marquages ($SCG_{\subset}$)

Marquages + états + traces (SSCG [BV03])

Marquages + états ($SSCG_{\subset}$)

Marquages + états + traces + branchements (CTL^*) (ASCG [BV03])

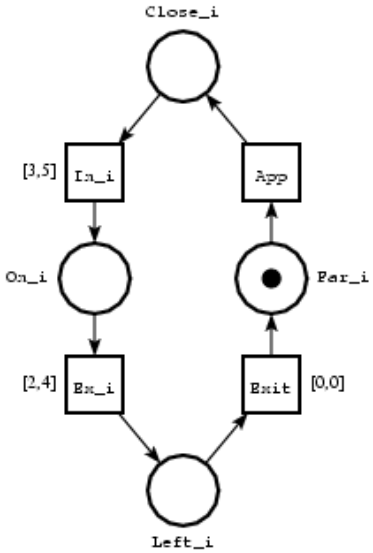
Aussi :

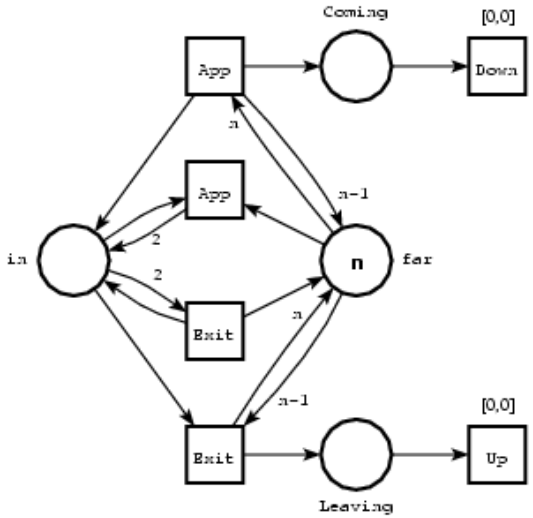
Espaces d'états en temps discret

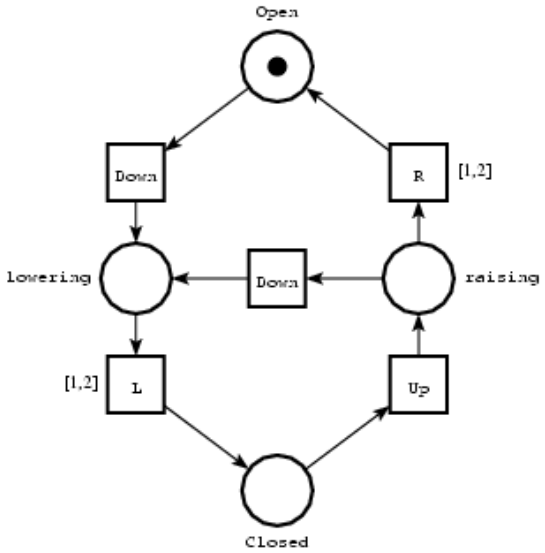
Méthodes symboliques (BDD, préservent Marquages)

Théorème : Abstractions finies ssi réseau est borné

Passage à niveau



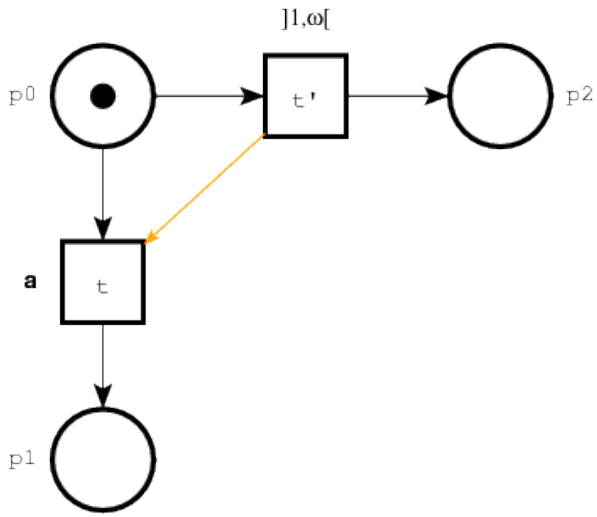




		M tina -M	*M* + *LTL* tina -W	*E* tina -E	*E* + *LTL* tina -S	*E* + *CTL* tina -A	*M* + *LTL* (*discret*) tina -D	*M* + *LTL* (*discret*) tina -F
(1 *train*)	*Classes*	10	11	10	11	12	13	23
	Transitions	13	14	13	14	16	27	36
(2 *trains*)	*Classes*	37	123	41	141	195	116	382
	Transitions	74	218	82	254	849	198	373
(3 *trains*)	*Classes*	172	3101	232	5051	6973	1550	2280
	Transitions	492	7754	672	13019	49818	5823	5275
(4 *trains*)	*Classes*	1175	134501	1807	351271	356940	22268	28830
	Transitions	4534	436896	7062	1193376	1447835	91256	81077
(5 *trains*)	*Classes*	10972	8557621	18052	35945411	23081275	313214	372264
	Transitions	53766	34337748	89166	151908273	279572133	1397517	1245355
(6 *trains*)	*Classes*	128115	697913229	217647	?	?	4299116	4830558
	Transitions	760538	3334109864	1297730	?	?	20886774	18833697

Priorités

Réseaux Temporels + Priorités (PrTPN)



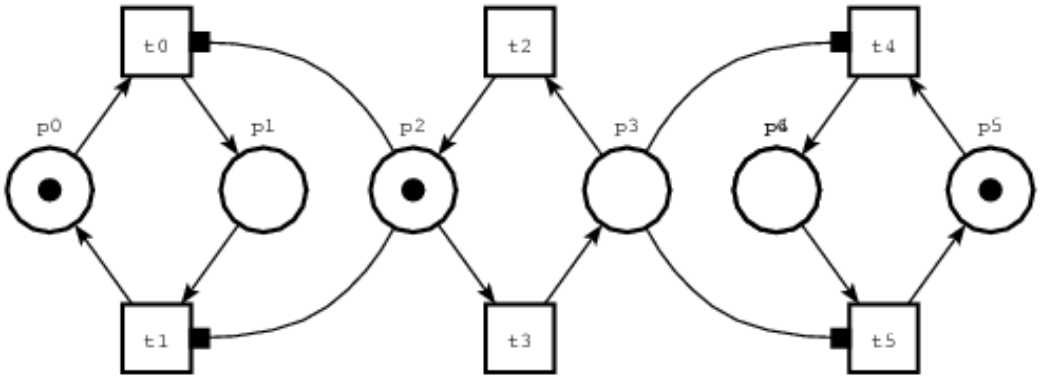
Priorités augmentent l'expressivité des TPN (PrTPN bornés $\approx$ TA)
Pour PrTPN, l'écoulement du temps peut rendre une transition non tirable

Abstractions

Classes d'états restent applicables (SSCG, ASCG, mais pas SCG)

Suspension et reprise de transitions

Réseaux Temporels à Chronomètres (SwTPN)



Une transition sensibilisée peut être *Active* ou *Suspendue*

Applications : systèmes ordonnancés, préemption temporelle

Abstractions

Graphes de classes adaptables, MAIS accessibilité indécidable ...

Surapproximations fournissent des conditions suffisantes ou nécessaires

Prise en compte des données

Time Transition System =

Systèmes de Keller

marquages $\Rightarrow$ états (vecteurs d'entiers)
transitions "additives" $\Rightarrow$ transitions arbitraires

\+ **contraintes temporelles à la TPN**

On perd : décidabilité du caractère borné

en Tina :

format tts = TPN + traitement de données synchronisé (en C)

Abstractions

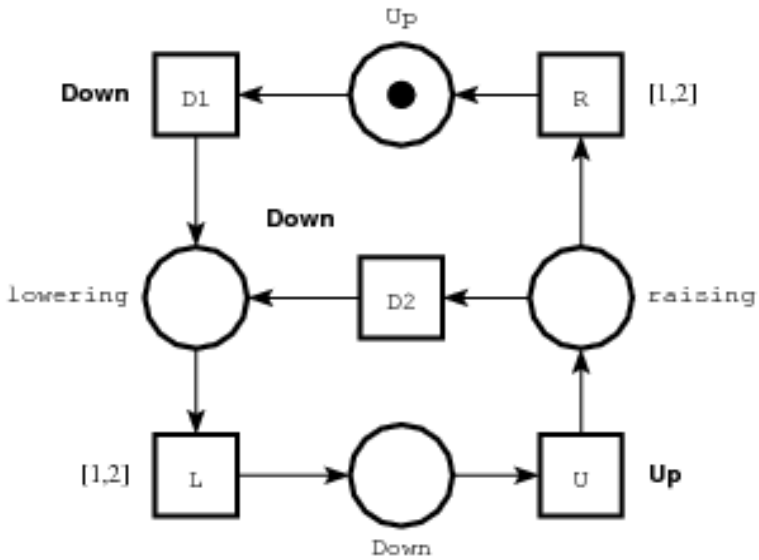
Méthode des classes d'états reste applicable

Descriptions de haut niveau — FIACRE

cf Projets COTRE, TOPCASED, OpenEmbeDD

```
process barrier_p [Down,Lower,Up,Raise: sync] is
  states up lowering down raising
  init up
  from up       Down; to lowering
  from lowering Lower; to down
  from down     Up; to raising
  from raising  select Raise; to up
                []     Down; to lowering
                end

component barrier [Down,Up: sync] is
  port Lower, Raise : none in [1,2]
  barrier_p [Down, Lower, Up, Raise]
```



Le vérificateur SELT

Formules

S/E-LTL + arithmétique, e.g.

$\Box(t1 \Rightarrow \Diamond(p2 \geq p3 + p4 \vee p6))$

Contre exemples Abrégés

```
- [] (t1 => <> t4);
FALSE
  state 0: p1 p2*2
  -t1 ... (preserving - t4 /\ t1)->
* [accepting] state 12: p3 p4 p5
  -t5 ... (preserving - t4)->
  state 12: p3 p4 p5
```

Peuvent être rejoués dans le simulateur Tina

Prospective

Descriptions de haut niveau

- Poursuite intégration Fiacre

Passage à l'échelle

- Parallélisation Exploration et vérification
  - Démonstrateur Mercury multi-core, mémoire partagée (NUMA) : speedup 8 à 10 sur machine 8*2 cores
- Vérification probabiliste parallèle

Optimisation de modèles

- Simplification de modèles (réductions, etc)
- Prise en compte informations spécifiques (symétries, invariants)
- Interprétation abstraite, pour systèmes infinis (Fiacre)

Liens

Outils

`http://www.laas.fr/tina`

`http://www.laas.fr/fiacre`

Références

`http://www.laas.fr/tina/papers.php`

`http://www.laas.fr/fiacre/papers.php`

Fiacre

Format Intermédiaire pour l'Assemblage de Composants Répartis

Bernard Berthomieu Silvano Dal Zilio
Didier Le Botlan François Vernadat

Vertics / LAAS / CNRS / INSA

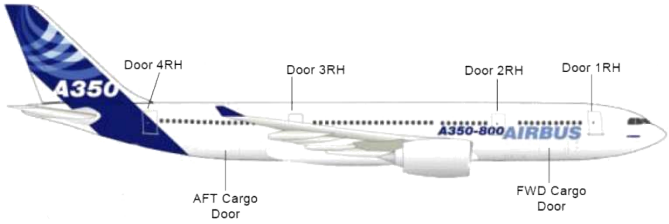
Journée AFSEC Concurrence – Cachan, 19 Novembre 2013

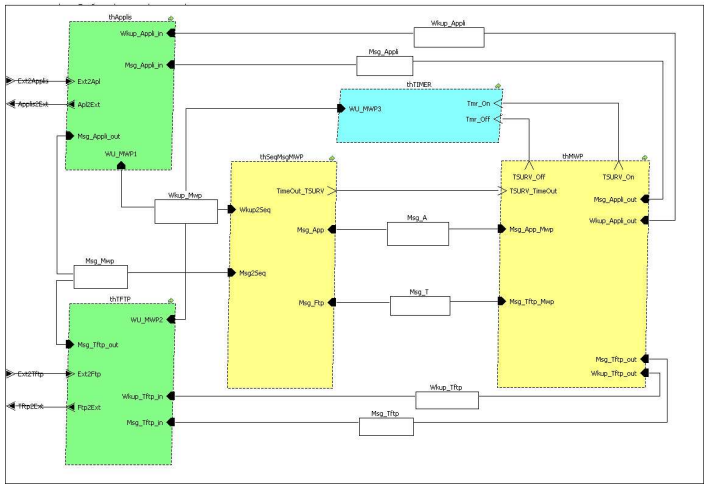
Justification

- Un langage d'entrée pour le format TTS
- Un langage de haut-niveau pour définir les modèles
  - déclaration hierarchique et compositionnelle
  - manipulation des données
- Un langage pour l'ingénierie système

Ingénierie Système

software + protocols +
hardware + …



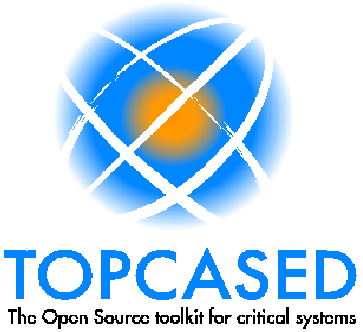


manipule un grand nombre
de “*notations*” différentes …

System Eng. ≈ Software Eng. avec des
préoccupations différentes: fiabilité, LTS; …

www.topcased.org

- Toolkit in OPen source for Critical Application and SystEm



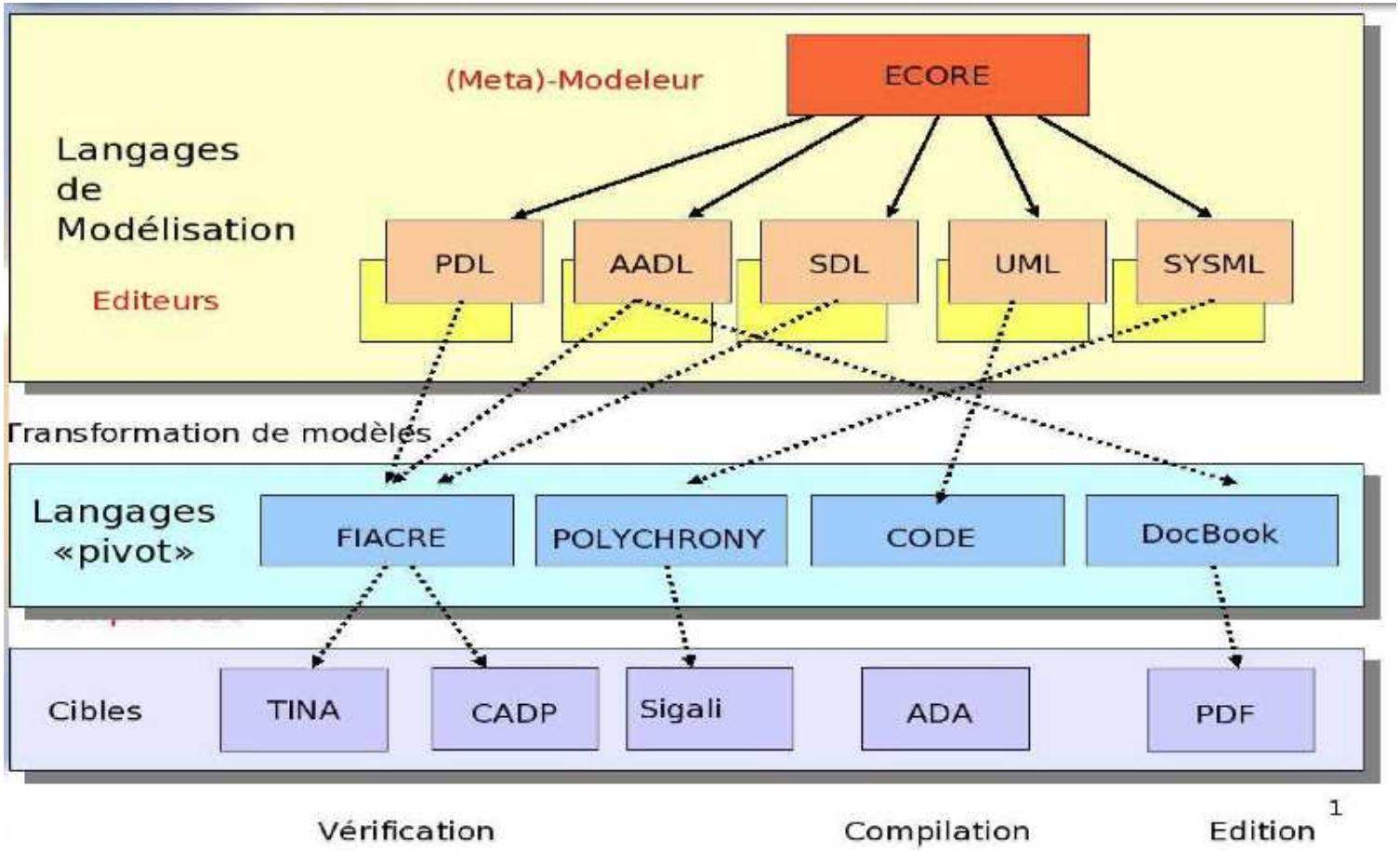
- *Eclipse for system-engineering* !?

PolarSys

Open Source tools for the development of embedded systems

Eclipse IWG
www.polarsys.org

Intégration Eclipse / Topcased



langage de vérification

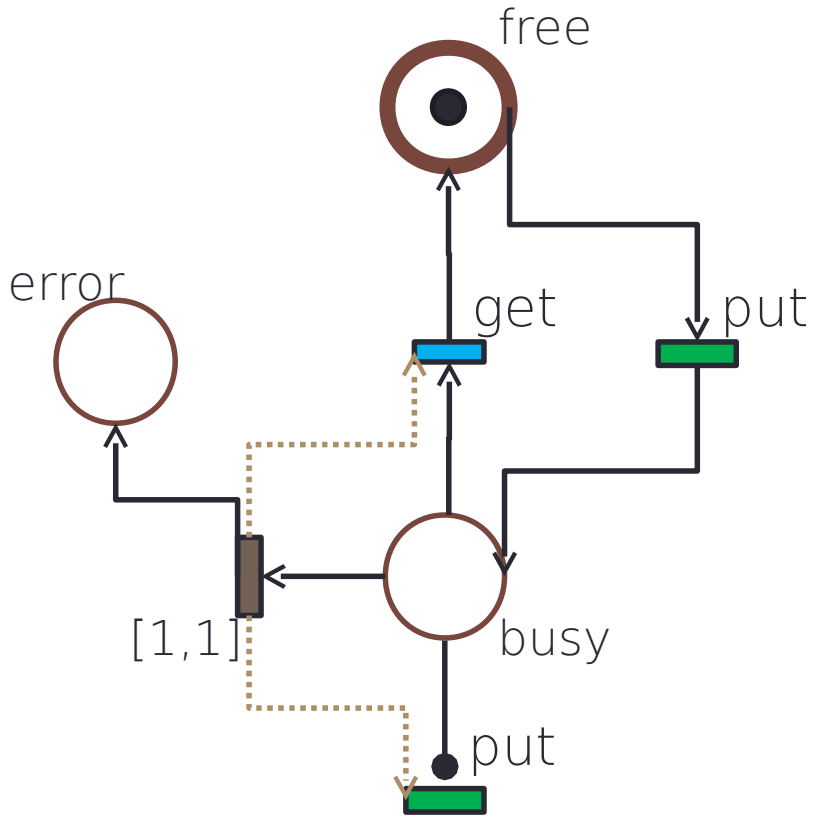
Language formel de spécification

- Basé sur la notion de *processus*
  - gestion explicites des états
  - transitions symboliques (structures de contrôle classiques)
  - Riches types de données
  - Communications (variables partagées ET messages)

- Définition hierarchique: *composants*
  - défini les interactions entre sous-composants
  - unité de définition des contraintes temporelles
  - contraintes de priorité sur interactions

Un exemple simple

```
process lbuf [put : none, get : none] is
    states busy, free, error
    from free put ; to busy
    from busy
        select
            put ; loop
        []  get ; to free
        unless wait [1,1] ; to
        error
        end
```



Aspects du langage

- État = « marquage » + données + temps
- Typage statique fort
- Langage d'expressions complexes pour définir les (macros-) transitions
- Étiquetage des transitions et priorités

Fiacre: typage statique fort

- **types de bases :** int, nat, bool
- **intervalles :** type byte is 0..255
- **unions et énumérations :**

  type color is union red | green | blue end

  type val is union Nat of nat | Bool of bool end
- **enregistrements :**

  type size is record width : nat, height : nat end
- **Tableaux et files de taille fixe :**

  type table is array 10 of nat end

  type fifo is queue 5 of byte end

Fiacre: instructions

- Affectation déterministe et non-déterministe

  a[6].width = 1

  x = any where x < 3

- Conditionnelles:

  if C then E1 else E2 end

  while C do E end

  foreach C do E end

Fiacre: instructions (suite)

- **Choix non-déterministe:** `select E1 [] ... [] En end`
- **Séquence:** `E1 ; E2`
- **Communication:**

  `port` (synchro)

  `p!E1, ..., En` (émission)

  `p?X1, ..., Xn` **where** `E` (réception)

- **Continuation:** `to snext` **ou** `loop`

Fiacre: composants

- Comportement = composition + contraintes
  - La composition parallèle d'instances des processus et composants (hiérarchique)
  - Des priorités entre actions
  - Des contraintes temporelles sur les communications

- création d'instances et la mise en parallèle de processus

- paramétrés par: des ports de communication; variables (partagées ou par valeurs)

```
type id is 1..2
type lock is 0..2

process Proc(pid : id, &lock : lock) is

   states Init, WaitLock, SetLock, TestLock, TestCS, CS

   from Init      wait [1, ...[ ; to TestLock

   from TestLock  wait [0, 0]  ;
      if (lock = 0) then to SetLock else to Init end

   from SetLock   wait [0, 2] ; lock := pid ; to WaitLock

   from WaitLock  wait [3, ...[ ; to TestCS

   from TestCS    wait [0, 0] ;
      if (lock = pid) then to CS else to Init end

   from CS       wait [1, 1] ;
      lock := 0 ; to Init

component Main is
   var lock : lock := 0

   par Proc(1, &lock) || Proc(2, &lock) end
```

Fisher mutual exclusion protocol (2 instances)

langage d’assertion

Extensions

- Observables et propriétés
  - riche langage d'observables (state, value, event, ⋯)
  - instrumentation fiacre automatique
  - langage d'assertion à base de patterns

Propriétés

- Propriétés génerales:

  deadlockfree

  infinitelyoften o

- Pattern présence

  present o

  present o after o'

  present o before o'

  present o between o1 and o2

  present o after o1 until o2

Propriétés: suite

- **Absence:** absent o
- **Response:** o1 leadsto o2
- **Universality:** always o
- **Precedence:** o1 precedes o2
- **Composition:** P and Q not P P and Q

Propriétés temporisées

- Pattern Presence

  present o within [a, b]

  present o lasting d

  present o after o' within [a, b]

- Pattern Leadsto

- ...

Applications

Industrial Use Cases

Mise en œuvre de l'approche Fiacre/Tina dans des projets industriels

Bernard Berthomieu Silvano Dal Zilio
Didier Le Botlan François Vernadat

Vertics / LAAS / CNRS / INSA

École d'Été Temps Réel (ETR) — Août 2013

23

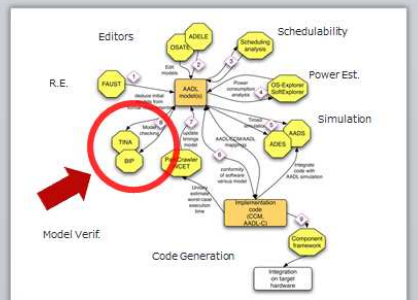
AADL

24

Exemple 1: AADL

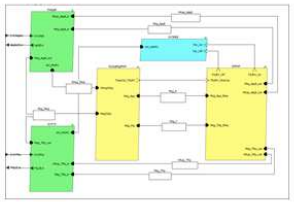
- Use of AADL for real-time embedded software architecture
- Evaluate capabilities of the language and of the tooling: modelers, code/doc generators, checkers, …

25



26

AADL Graphical Syntax

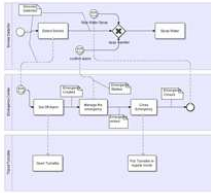


BPEL

28

Exemple 2: BPEL / BPMN

- Langages de description de workflow



29

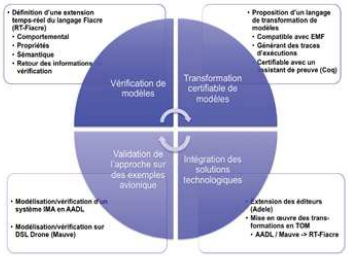
SOA & syst. embarqués

30

IDM

Exemple 3: Ingénierie des Modèles

- Modèles *(à la UML)*, transformation de modèles et model-checking
- **Objectifs**: chaîne de transformation qualifiable entre langages métier de description de systèmes et outils de model-checking.
- **Besoins**: extension du langage intermédiaire (RT-Fiacre) et des technologies de transformation
- **Use cases**: AADL, Mauve (DSL Drone)



Liens

- http://projects.laas.fr/fiacre/

Projets :

ANR OpenEmbeDD: http://openembedd.org
ITEA Spices: http://www.spices-itea.org
ANR Itemis : http://itemis-anr.org
FNRAE Quarteft: http://quarteft.loria.fr
ARTEMIS Cesar : http://cesarproject.eu/
ITEA2 openETCS : http://openetcs.org/